Hochiminh City University of Technology

Computer Science and Engineering

[CO1027] - Fundamentals of C++ Programming

Control Flow – Loop

Lecturer: Duc Dung Nguyen
Credits: 3

Today's outline

❖ Loop statements: while, for, do-while

❖ Structure programming

Loop statements

while statement

❖ Why do we need iterations?

    ❖ Waiting for something to happen

    ❖ Operate on several objects

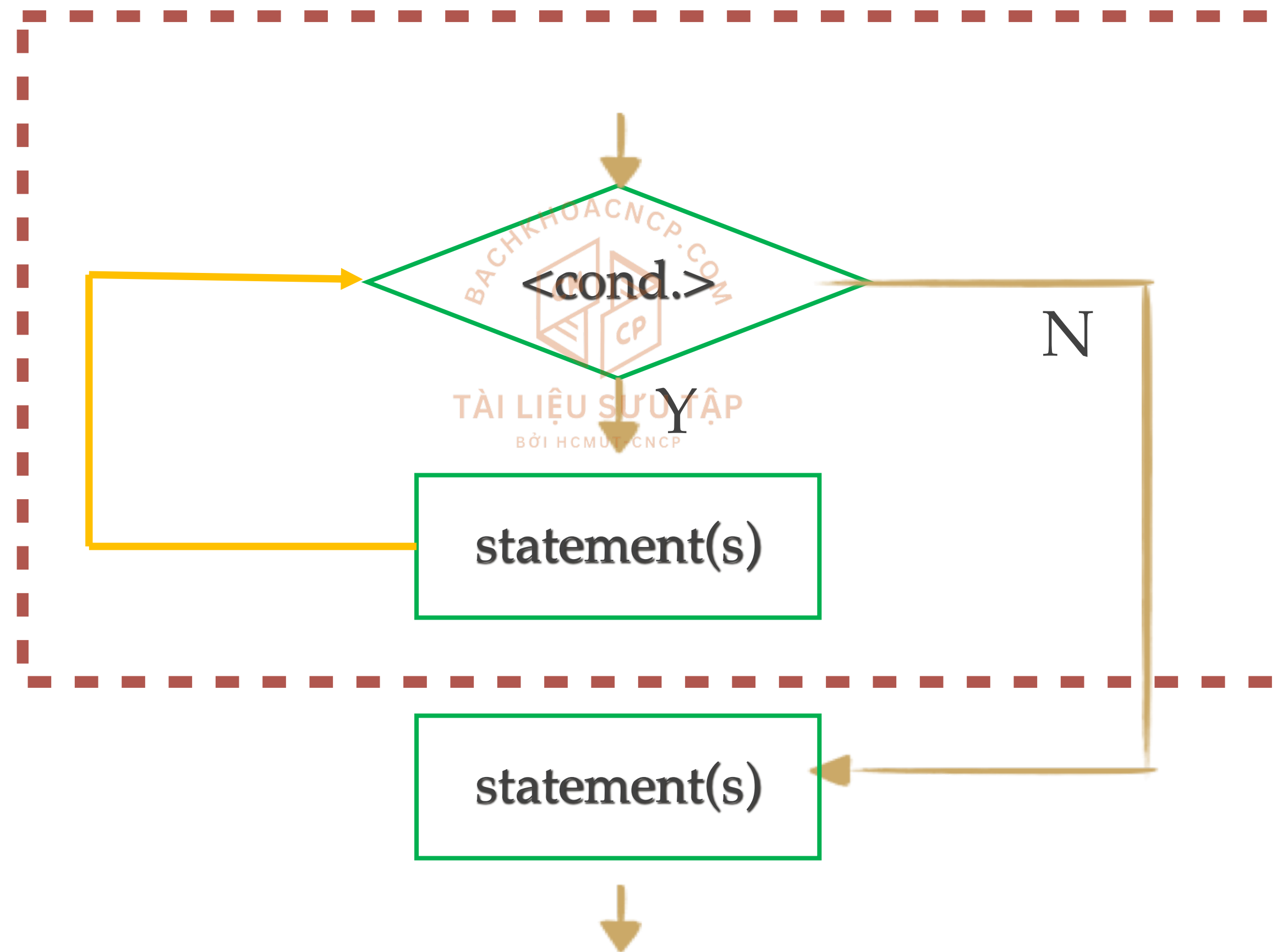   ❖ List, array of objects

        ❖ String

while loop

❖ Syntax:

  ❖ `while (<condition>) <statement>;`

  ❖ ```
while (<condition>) {
    <statements>;
}
```

# while loop

❖ Flowchart

# Example

```cpp
#include<iostream>
using namespace std;

int main() {
    int counter = 0;
    while (counter < 10) {
        cout << counter << " ";
        counter++;
    }
    cout << endl;
    return 0;
}
```

Do-while loop

- ❖ Syntax:

  - ❖ `do <statement> while (<condition>);`

  - ❖ `do {`
    `    <statements>;`
    `} while (<condition>);`

Do–while loop

❖ Flowchart

Example

```cpp
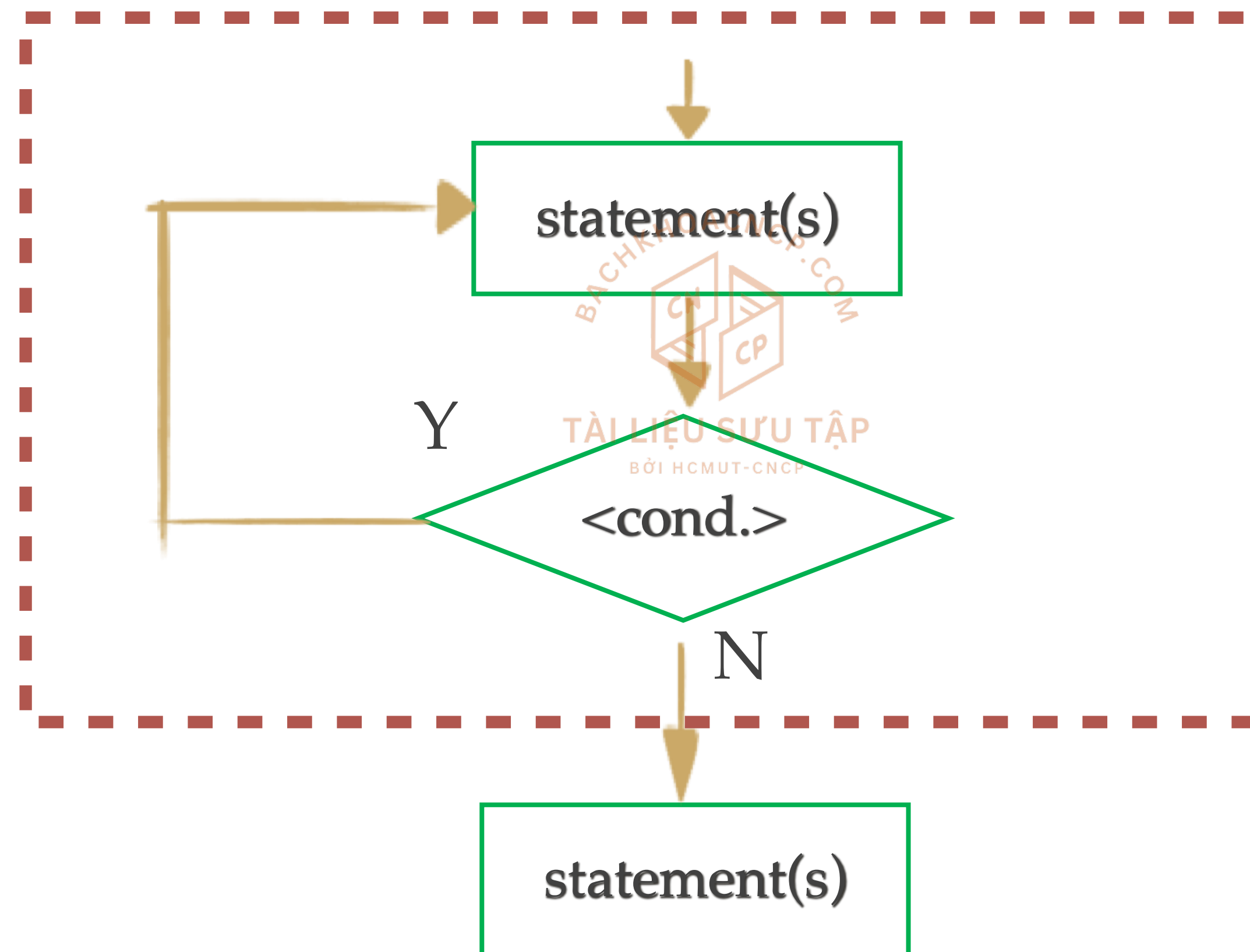#include<iostream>
using namespace std;

int main() {
    int counter = 0;
    do {
        cout << counter << " ";
        counter++;
    } while (counter < 10);
    cout << endl;
    return 0;
}
```

while statement

- Note:

  - Remember to initialize variables in the condition expression before entering the while statement (at least you know what will happen when you check the condition).

  - Do not forget stopping condition.

  - Take care of counters.

for statement

❖ Why do you need for statement?

   ❖ Just another way to write iteration/loop structure!

   ❖ Counting is a frequent activity

   ❖ for: a specialised loop that package the following tasks in a statement

      ❖ Initialise a counter variable

      ❖ Modify the counter

      ❖ Check complete condition

for statement

❖ for loop:

- ❖ for (<initialization>; <condition>; <modification>) <statement>;

- ❖ for (<initialization>; <condition>; <modification>) {
    <statements>;
  }

for statement

❖ Flowchart

for statement

❖ Initialization: set value for the counter

    ❖ Declare one or many counters (same type) and init them at once

    ❖ Initialize many counters if needed

❖ Condition: a boolean expression that must be evaluated at each loop

❖ Modification: change value of the counter at each loop

Example

```cpp
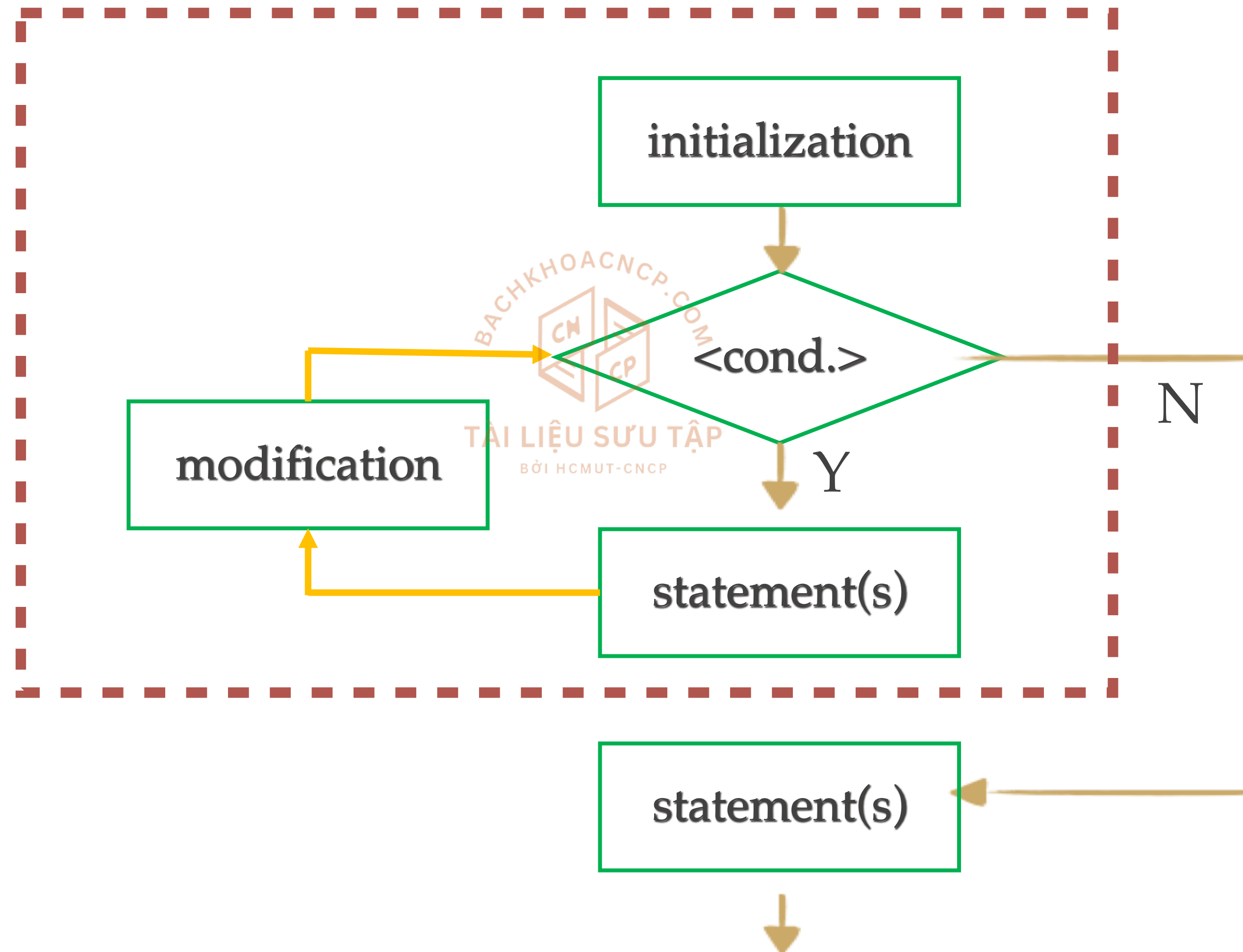#include<iostream>
using namespace std;

int main() {
    for (int i = 0; i < 10; i++)
        cout << i << " ";

    cout << endl;
    return 0;
}
```

for statement

❖ Note that initialization and modification can contain multiple statements separated by commas.

```cpp
#include <iostream>
using namespace std;

int main() {
    int i, j;
    for (i = 5, j = 10; i + j < 20; i++, j++) {
        cout << "i + j = " << (i + j) << '\n';
    }
    return 0;
}
```

Infinite loops

```
while (100) {

}

while (true) {

}

do {

} while (-20);


for (;;) {

}
```

Exist loops

❖ The two most commonly used are:

  ❖ break: will end the loop and begin executing the first statement that comes AFTER the end of the loop.

  ❖ continue: force the next iteration to be executed.

Example

```cpp
#include <iostream>
using namespace std;

int main() {
    int count;
    for (count = 1; count <= 10; count++) {
        if (count == 5)
            break;
        cout << count << " ";
    }

    cout << "\nBroke out of loop at count = " << count << endl;
    return 0;
}
```

Example

```cpp
#include <iostream>
using namespace std;

int main() {
    int count;
    for (count = 1; count <= 10; count++) {
        if (count == 5)
            continue;
        cout << count << " ";
    }

    cout << "\nUsed continue to skip printing 5" << endl;
    return 0;
}
```

Nested loop

❖ A loop can be nested inside a loop.

❖
```
while (<condition 1>) {
    <statements>;
    while (<condition 2>) {
        <statements>;
        while (<condition 3>);
    }
    <statements>;
}
```

Example

```cpp
#include <iostream>
using namespace std;

int main() {
  int i, j;

  for (i = 2; i<100; i++) {
    for (j = 2; j <= (i / j); j++)
      if (!(i%j)) break; // if factor found, not prime
    if (j >(i / j)) cout << i << " is prime\n";
  }

  return 0;
}
```

Structure programming

Structure programming

❖ Definition: a programming paradigm aimed at improving the clarity, quality and development time of a computer program by making extensive use of subroutines, block structures and for/while loops

❖ Structured programming languages: ALGOL, Pascal, PL/I, Ada, **C/C++**, etc.

Structure programming

❖ Loop and array

  ❖ Loop is good for performing operations on arrays, strings.

  ❖ "while", "do-while", "for" are exchangeable.

  ❖ Fixed size data should be processed using finite loops.

Problem solving – example

❖ Input and draw the following figure in terminal:

  ❖ Input: N (number of lines)

  ❖ Output: (in case N = 5)

```
        *
      *   *
    *   *   *
  *   *   *   *
*   *   *   *   *
```

Problem solving – example

❖ Input and draw the following figure in terminal:

   ❖ Input: N (number of lines)

   ❖ Output: (in case N = 5)
```
*           *
**         **
***       ***
****     ****
********
```

Problem solving – example

❖ Input and draw the following figure in terminal:

   ❖ Input: N (number of lines)

   ❖ Output: (in case N = 5)
```
*           *
**         **
 *  *    *  *
 *   *  *    *
 *     *     *
```

Summarise

❖ Understand loop structures: while, do-while , for

❖ Implements algorithms with loops